Applied Mathematical Finance and its Object Oriented Implementation.

Project:

Model Extension to Calibrate LMM

to Swaption Implied Volatility Smile

Christian P. Fries
email@christian-fries.de

June 12, 2013

Version 1.0

Abstract

The following are the exercises of a project consisting of the extension of a LIBOR market model and its implementation.

The project is conducted in the LMU quantLab, but may be prepared at home.

Support is provided by

Christian Fries, email@christian-fries.de

Juan Miguel Montes, montes.jmiguel@gmail.com

Resources and further details can be found at
`http://www.christian-fries.de/finmath/lecture13.2/project/`

Contents

1 Introduction

The mid term project consists of the extension of a LIBOR market model and its implementation. This part may be performed in groups of up to 3 students.

1.1 Aim of the Project

We will first compare the properties of a LIBOR market model to given market data. This motivates extension of the model. Our aim is to utilize object orientation to implement this modification. The students will collaborate using the subversion revision control system (students obtain write access to the project repository).

Having checked that the model extension is indeed an improvement, we motivate the derivation of a analytic approximation for a fast calibration.

1.2 Evaluation of the Project Results

To successfully pass the review of the project a short presentation of a part of the solution has to be performed (parts will be distributed randomly) together with answering a set of project related questions.

Note: The implementation part of the project can be solved very elegantly using object oriented implementation techniques requiring only some 20 lines of new code. We encourage you to discuss your ideas during the solution in order to improve you solution.

2 Motivation

2.1 Task: Explore the (Implied) Volatility Smile of Market Data and Model Values

Exercise 1 (Implied Volatility of a standard LIBOR Market Model): The spreadsheet `MarketDataExample.ods` gives some market data for swaptions.[1] It also gives the *implied volatility smile* for 5-in-10 swaptions, i.e., the implied Black volatility for options on a 10 years swap with option maturity in a 5 years for different strike swap rates.

1. Implement a small test which calibrates a LIBOR market model to the (ATM) swaption volatilities contained in `MarketDataExample.ods` and compare the model implied volatility of 5-in-10 swaptions (with different strikes) to the market data. Create a plot of the data. What do you observe?[2]

2. Explore the spread sheet spreadsheet `LIBORMarketModel.ods` and check that it uses the market data in`MarketDataExample.ods`, if not configure it accordingly. Check the implied volatility smile using this sheet.

[1] Resources, like spreadsheets, are available at `http://www.christian-fries.de/finmath/lecture2013.2/project`.

[2] Note: This exercise gives the motivation for the following (more demanding) exercises.

3 Model Extension

3.1 Task: Modify the Code Base towards a Normal Model

Exercise 2 (Modify the Code Base towards a Normal Model): The standard LIBOR market model in general cannot calibrate to swaptions having the same option maturity and swap tenor, but different strike. In other words: the implied volatility smile cannot be changed. To improve the calibration, extend the model with a local volatility model. To prepare this step change the numerical implementation of `LIBORMarketModel.java`.

The implementation of `LIBORMarketModel.java` specifies the model $L_i(t) = \exp(Y_i(t))$ with

$$dY = \mu^Y(t, L)\mathrm{d}t + \sigma(t, L) \cdot \mathrm{d}W(t)$$

Implement a new version[3] which specifies the model $L_i(t) = Z_i(t)$ with

$$dZ = \mu^Z(t, L)\mathrm{d}t + \sigma^Z(t, L) \cdot \mathrm{d}W(t).$$

3.2 Task: Create a local volatility model)

Exercise 3 (Creation of a Local Volatility Model): Create a local volatility model g(t,L) such that

$$\sigma^Z(t, L) := g(t, L)\sigma^Y(t, L)$$

with $g(t, L) = aL + (1 - a)$.

Having implemented this, modify the spreadsheet `LIBORMarketModel.ods` to use your new model and check that

- $a = 1$ gives almost the same calibration as the (plain) `LIBORMarketModel.java` (it is just a different numerical implementation).
- $a = 0.1$ allows to fit to the swaption smile form Exercise 1 much better. Due to the lack of an analytic calibration formula, this calibration has to be done via Monte-Carlo which can take very long.[4]

Document the calibration accuracy for the different models using spreadsheets. *Note: At this point you have extend the model by an additional parameter which allow much better calibration of the volatility smile.*

[3] Create a copy, e.g., `LIBORMarketModelNormal.java` and apply the required modifications.

[4] For $a \neq 1$ you need to change the calibration products used in the method `getCalibrationItems` of your new class `LIBORMarketModelNormal` to the general Monte-Carlo products by setting `isUseAnalyticApproximation` to `false`. This is necessary, because the analytic approximation is only valid for $a = 1$. This will be tackled in the next exercise.

4 Fast Calibration

4.1 Task (advanced / facultative): Derive and Implement an Approximate Analytic Valuation for Fast Calibration

Exercise 4 (Derive and Implement an Approximate Analytic Valuation for Fast Calibration): Bonus Exercise (advanced)

1. Derive an analytic calibration for the displaced diffusion LIBOR market model, i.e., the model from the previous exercise.
2. implement the analytic calibration formula.

References

[1] FRIES, CHRISTIAN P.: LIBOR Market Model. Object oriented reference implementation.
http://www.finmath.net/topics/libormarketmodel.

8 pages. 0 figures. 0 tables.